

D3.6 – DOCUMENTATION AND SOFTWARE ON NEW METHODS, TOOLS AND MECHANISMS FOR RELEASE 3 OF THE PORTAL

Grant Agreement	676547
Project Acronym	CoeGSS
Project Title	Centre of Excellence for Global Systems Science
Topic	EINFRA-5-2015
Project website	http://www.coegss-project.eu
Start Date of project	October 1, 2015
Duration	36 months
Deliverable due date	31.01.2018
Actual date of submission	31.01.2018
Dissemination level	Public
Nature	Report
Version	1.0
Work Package	WP3
Lead beneficiary	HLRS
Responsible scientist/administrator	Marcin Lawenda
Contributor(s)	Marcin Lawenda, Francisco Javier Nieto De Santos, Burak Karaboga, Ingo Brauckhoff, Michał Pałka, Johan Lodin, Fabio Saracco, Andrea Rivetti, Wolfgang Schotte, Piotr Dzierżak, Steffen Fuerst, Sergiy Gogolenko

Internal reviewers	Sarah Wolf Tiziano Squartini
Keywords	portal, global system sciences, HPC, centre of excellence, CoeGSS, data management, synthetic population, synthetic network, data analysis, ABMS, CKAN, Moodle, AskBot, LDAP
Total number of pages:	47

Copyright (c) 2016 Members of the CoeGSS Project.



The CoeGSS (“Centre of Excellence for Global Systems Science”) project is funded by the European Union. For more information on the project please see the website [http:// http://coegss-project.eu/](http://coegss-project.eu/)

The information contained in this document represents the views of the CoeGSS as of the date they are published. The CoeGSS does not guarantee that any information contained herein is error-free, or up to date.

THE CoeGSS MAKES NO WARRANTIES, EXPRESS, IMPLIED, OR STATUTORY, BY PUBLISHING THIS DOCUMENT.

Version History

	Name	Partner	Date
From	Marcin Lawenda	PSNC	December 5th, 2017
Contribution from	Francisco Javier Nieto De Santos, Burak Karaboga, Ingo Brauckhoff, Michał Pałka, Johan Lodin, Fabio Saracco, Andrea Rivetti, Wolfgang Schotte, Piotr Dzierżak, Steffen Fuerst, Sergiy Gogolenko	ATOS, Chalmers, IMT, ISI, HLRS, PSNC, GCF	January 19 th , 2018
Version 0.65	for internal review	PSNC	January 23rd, 2018
Reviewed by	Sarah Wolf Tiziano Squartini	GCF IMT	January 29th, 2018
Version 1.0 for submission	Marcin Lawenda	PSNC	January 30th, 2018
Approved by	Coordinator	UP	January 31st, 2018

Abstract

This document is foreseen as a continuation of the previous deliverable D3.5 where achievements were addressed in implementing services specified in the deliverable D3.2 as far as they are realized to be integrated in release 2 of the portal, planned at month 20, the overall Centre's workflow.

This document is the continuation of deliverable D3.5 which reported achievements in the implementation of services specified in deliverable D3.2 and their integration in release 2 of the portal planned at month 20 of the centre's workflow.

In this deliverable, we are presenting the progress in the development of tools which are intended to be integrated in release 3 of the CoeGSS portal as well as mechanisms that will bind portal and HPC functionality.

It should be mentioned that work on the implementation is still in progress, so some solutions are presented in their current state. In these cases, further investigation and development is expected.

Table of Contents

Abstract	3
Glossary	5
1 Introduction.....	7
2 Current state of the Portal	8
3 Documentation of operating components	10
4 Software integration	14
5 Software outlook.....	34
6 Summary	42
References.....	43
List of tables	45
List of figures	46

Glossary

3D	Three-dimensional space.
ABM	Agent-Based Model
ABMS	Agent-Based Modeling and Simulation
Apache Spark	an open-source cluster-computing framework
API	Application Programming Interface
AskBot	a popular open source Q&A Internet forum
CAVE	Cave Automatic Virtual Environment (an immersive virtual reality environment)
CKAN	Comprehensive Knowledge Archive Network
CoE	Centre of Excellence
CoeGSS	Centre of Excellence for Global Systems Science
COVISE	Collaborative Visualization and Simulation Environment
CPU	Central Processing Unit
CRB	COVISE Request Broker
CSS	Cascading Style Sheets
CSV	Comma-Separated Values file format
Cuda	a parallel computing platform and API model created by Nvidia
D	Deliverable
DCAT	Data Catalog Vocabulary
DDR (DDR SDRAM)	Double Data Rate Synchronous Dynamic Random-Access Memory
Django	a free and open-source web framework, written in Python
DSL	Domain-Specific Language
EC	European Commission
Eclipse CDT	Eclipse C/C++ Development Tooling
GB	Gigabyte
GB/s	Gigabytes per second
GDDR	Graphics DDR SDRAM
GHz	Gigahertz
GIS	Geographic Information System
GitHub	a web-based VCSs repository and Internet hosting service
GPU	Graphics Processing Unit
GSS	Global Systems Science
HDF5	Hierarchical Data Format, version 5
HPC	High Performance Computing

HTTP	Hypertext Transfer Protocol
I/O	Input/Output
IDE	Integrated Development Environment
IP	Internet Protocol
IPF (IPFP)	Iterative Proportional Fitting Procedure
LDAP	Lightweight Directory Access Protocol
M	month
Moodle	a free and open-source software learning management system
OpenCOVER	Open COVISE Virtual Environment (an integral part of the COVISE visualization and simulation environment)
PBS	Portable Batch System (computer software that performs job scheduling)
PDF	Portable Document Format
PostgresSQL (Postgres)	an object-relational database with an emphasis on extensibility and standards compliance
Q&A software (FAQ Service)	a Web service that attempts to answer questions asked by users
RAID	Redundant Array of Independent Disks (a data storage virtualization technology)
RAM	Random-Access Memory
RDF	Resource Description Framework (a family of W3C specifications designed as a metadata model)
SCM	Software Configuration Management
SI	Synthetic Information
SLURM (Slurm)	Simple Linux Utility for Resource Management (workload manager, job scheduler)
SM	Streaming Multiprocessor
SQL	Structured Query Language
SSH	Secure Shell (a cryptographic network protocol)
SSO	Single Sign-On
TCP	Transmission Control Protocol
TB	Terabyte
VCS	Version Control System
VNC	Virtual Network Computing
VR	Virtual Reality
W3C	World Wide Web Consortium
WP	Work Package
XLS	Excel Binary File Format

1 Introduction

With this deliverable we are providing the knowledge and arrangements we gained and developed since the previous document, D3.5, which should be taken as complementary when reading of this deliverable.

Previously we focused on the description of tools strictly designed to provide functionality in the portal like Moodle and AskBot as well as interfacing capabilities which by their nature lie beneath these services, like data management operated by the CKAN server.

In the meanwhile, we carried out many analyses and tests to advance service functionality and interface them with the portal (chapter 2).

As was stated in the previous document, the portal has a single entry where CoeGSS stakeholders may run provided services. This assumption has a significant impact on design and implementation of the services.

The overall goal of the services developed by WP3 is to use the potential and possibilities of HPC and HPDA infrastructure. That is why these applications must be run directly on machines where a multithreading approach and high throughput processing are feasible.

Portal functionality must guarantee handling HPC applications directly from its level. In order to couple these two environments (portal and HPC), special solutions described in chapter 3 are envisaged.

In the portal, two types of interactions can be distinguished: batch and interactive. The batch job definition and submission are sufficient for most of the applications like Synthetic Populations (chapter 4.1) and Network Reconstruction (chapter 4.2). A special approach is proposed for visualisation where interactive access is required, a dedicated solution based on the COVISE software is presented in chapter 4.3. Data manipulation solutions leading to a common data format are explained in chapter 4.4.

In chapter 5.1 we discuss the progress in adaptation and development of ABMS tools like Pandora and AMOS. We are concluding our considerations with chapter 5.2 where data analytics approaches are presented for undemanding and demanding data exploration.

2 Current state of the Portal

In order to avoid duplication, the status of the CoeGSS portal will be briefly highlighted by summarizing the information already available in deliverables D5.1 [10], D5.2 [11], D5.9 [12], D5.10 [13] and D5.11 [20].

The second release of the CoeGSS portal has been deployed in month M20 of the project's timeline adding several new features, enhancements to the existing components, visual improvements and a new authentication and authorization mechanism on top of what has been described in D3.5.

The new components of the CoeGSS portal, as well as the components that have been improved with new features and enhancements, are as follows:

- **Frontend**

The new portal release introduced a single sign-on (SSO) mechanism provided by a Shibboleth 2.0 deployment that relies on the existing LDAP deployment for authentication [20]. The Frontend also features two new sub-components called Yellow Pages and Matchmaking Tool.

- The Yellow Pages provides a catalogue of organizations that have a relationship with CoeGSS, which can be browsed and searched by any user of the portal. The catalogue will be extended with experts and organization employees in the next iteration of the CoeGSS portal [20].
- The Matchmaking Tool offers several web pages where the user is provided the best possible set of contacts for the required information. The matchmaking is done by processing the user needs taken as input and the existing data in the Yellow Pages.

- **Data management**

The second portal release introduced two new third party CKAN extensions, ckanext-extractor and ckanext-oaimph. Also, two new custom CKAN extensions, Dataset Relationship Manager and Data Movement have been implemented:

- Dataset Relationship Manager enables the user to create and edit the relations between existing datasets.
- Data Movement allows the user to import data from outside sources (e.g. MIDAS database) to the CKAN server [20].

- **Training & User support**

The current version of the CoeGSS portal introduces a new look and feel for both of the components to match the visual style of CoeGSS [20].

- **Infrastructure management**

With the addition of the SSO mechanism, two new virtual machines had to be added to the existing infrastructure.

For the next releases of the CoeGSS portal, the main goal is to provide fully functional HPC-aaS features allowing the user to interact with the HPC providers for job submissions and job

management. Following this high priority goal for the future of the CoeGSS project, the tasks below were identified:

- Definition and creation of user roles and hierarchy
- Integration of this user hierarchy with other CoeGSS components (CKAN, Moodle, LDAP etc.)
- Content creation for the Community and Training components
- Functional enhancements of the existing components
- Visual improvements of the existing components

3 Documentation of operating components

This section documents solutions on interoperation of portal and HPC capabilities. We also discuss unification of the access as well as user support features.

3.1 Portal – HPC interoperability

One of the main purposes of CoeGSS is to support the usage of HPC technologies and, therefore, it is necessary that the access to HPC resources and information will be easy. Such interaction is centralized through the Portal with a component which deals with the job submission functionality and with monitoring mechanisms that can also be used for accounting purposes.

The job submission requires some specification of the kind of resources and other particularities of the software to be run. Moreover, when doing simulations, these follow a workflow already defined in CoeGSS, which requires using several tools in order to execute a complete simulation (data pre-processing, synthetic population generation, agent's simulation, big data analysis, etc...). Such workflows need somehow to be described, so the job submission system will be aware of the dependencies between tasks and of the required data movements.

The component in the Portal is based on Cloudify¹ and it uses TOSCA² as input for defining the workflows. Therefore, each complete simulation should provide a TOSCA file describing the tasks to be executed. In fact, we plan to have examples of TOSCA tasks for each of the tools developed in CoeGSS (i.e. synthetic population creation tool, network reconstruction tool, etc.), so it will be easier to 'compose' the workflows with chunks representing each step.

The jobs submission component is able to submit jobs to SLURM, and there is an ongoing implementation in order to support PBS/Torque. It is able to start, stop and destroy jobs and it can also retrieve some monitoring information, such as the status of a job, the status of the queue and status of the HPC resources. Such information can be used for performing a better selection of the HPC centre in which end users want to deploy their jobs.

It would be possible to perform applications monitoring while they are running, but that would require an instrumentation of the applications or an agreement on the way to generate logs, both assuming a performance loss, but enabling interactive simulations.

Finally, accounting is possible by enabling an interaction between the accounting systems of the different HPC centres involved in the project (HLRS and PSNC). Such interaction is not yet defined and it will not be available for the moment.

¹ Cloudify <http://cloudify.co/>

² Topology and Orchestration Specification for Cloud Applications <http://docs.oasis-open.org/tosca/TOSCA/v1.0/TOSCA-v1.0.html>

3.2 Access unification

3.2.1 Purpose, Design and Implementation

The initial portal design points out one of the main responsibilities of the Frontend component as providing a single point of access for all other components that are implemented in the context of CoeGSS. In the first Portal release, although the users' authentication information was centralized in LDAP, the login process had to be repeated for each individual service [20] which was not in-line with the design.

To achieve the design goal, a separate component, Shibboleth was selected with the purpose of providing the CoeGSS users a single sign-on mechanism, eliminating both the complexity of handling user sessions at the backend and the hassle of repeating the login process for each individual CoeGSS service for the user.

The component is composed of two sub-components, the Identity Provider (IdP) and the Service Provider (SP). IdP is deployed on its own virtual machine and each service provider resides on the same machine as the component that is integrated to the SSO mechanism (e.g. CKAN, Moodle, Askbot, and Frontend).

3.2.2 Use Cases

The SSO component works completely in the background except for taking control of the GUI for the login process, which is the only interaction with the user.

Once the user tries to access a restricted resource such as the user profile on the frontend or dataset creation on CKAN, he/she gets directed to the SP. At this point, the SP prepares an authentication request and sends it to the IdP. Here, IdP checks if the user has an existing session and if not, authenticates the user against LDAP by prompting for their username and password. Once the authentication is completed, the authentication response is sent to SP. After the validation of the response, SP creates a session for the user and sends him/her to the resource.

3.3 User support features of the portal

The design of the portal includes features for interacting and providing information about the services for both registered users and unregistered visitors of the website.

Interaction with registered users includes the implementation of a customer support system structured as a question and answer forum.

Requirements of the system include:

- a tagging system to provide flexible categorization
- a voting system to mark the most useful information
- easy filtering of all new issues submitted
- easy filtering of all issues closed with an answer

- easy filtering of all issues still open without an answer

The free software product AskBot is the tool chosen to implement such a system.

The AskBot system interfaces to the CoeGSS portal through LDAP allowing users with a valid portal login to authenticate on AskBot in order to raise questions and interact with the CoeGSS support staff.

The AskBot system has been configured with the following categories and sub-categories:

- Consultancy
 - Data-interpretation-visualisation-analytics
 - Model-building
 - Parallelisation
 - Simulation
- Matchmaking
- Portal
- Repository
- Training
 - How-to-get-support
- Yellow-Pages

The Categories are used to tag the requests for assistance so that each question can be efficiently redirected to the appropriate person in the second-level support staff.

The Categories will be reviewed and updated when new services are added to the portal.

The **AskBot system** is reachable from the main menu of the CoeGSS portal and its look and feel has been customized to match the general style of the CoeGSS website and portal. These customizations are packaged in a single CSS file that can be easily applied to new installations and that is carried forward between AskBot software updates.

The current implementation of the system is reachable at <http://support.coegss.hlr.de/>

Use Cases

The system provides information about problems encountered by the users of the services provided through the CoeGSS portal.

It is open to non-authenticated visitors to browse previous questions raised by the users and it requires authentication to raise new questions. Users authenticate with AskBot using the same credentials used to access the portal.

Non-authenticated users

For non-authenticated visitors, the existing questions (either answered or being discussed) can be browsed freely in order to find the solution or to follow the conversation which will eventually lead to the resolution of the issue.

Portal Users

For Portal Users, the same credential valid for the CoeGSS Portal can be used to access the AskBot system in order to raise a support request.

The system presents an input box marked as “search or ask your question” where the user starts entering the subject of his/her request, existing questions matching the keywords entered will appear along with the option of creating a new question.

The user can optionally tag the question giving an indication of the services impacted by the problem. Support staff will add/remove tags according to the areas where the issue needs to be investigated.

The user will come back to the portal to check the progress of his/her request and once support staff provides an answer, the user will be able to mark the solution provided as valid or to submit more information if the solution provided does not solve the problem.

Support Staff

Support staff will identify a new question and escalate the question as needed to receive information from the technical staff working on the impacted services.

Support staff acts as a “first line” support and logs into the AskBot system with “support staff” permissions.

AskBot provides a quick way to show the questions for which an answer has not been defined by selecting the “Unanswered” filter in the top menu.

Support staff provides follow-up information and makes sure the user tests the answer provided and marks the answer as correct or describes why the answer does not solve the problem in order to reach a satisfactory solution.

Support staff can also escalate the issue to technical points of contact for the service the problem is related to.

Each person acting as a technical point of contact acts as a “second line” support and has the specific expertise to troubleshoot one of the specific services provided by the portal. Their task is to provide information to the first-line support staff or take directly the responsibility to deal with the user through the AskBot system until a solution is reached.

4 Software integration

This section describes trial interfaces to the software components which must be integrated into release 3 of the portal like: Synthetic Populations, Network Reconstruction and visualization based on the COVISE application. This software was identified as an essential part of the general CoeGSS workflow in D3.2.

4.1 SynPop tool Web GUI

Synthetic populations are generated based on marginal distributions of population data and micro sample data. The generation procedure attempts to match the marginal distributions given as input by sampling from the micro sample data. The job specification consists of four parts:

- the vintage parameter,
- input data,
- output data,
- job description.

The vintage parameter is either a past timestamp or the label 'latest', which is equivalent to the current timestamp. The vintage parameter allows selecting older versions of the input files.

The input data identifies the datasets that are to be used by the generation procedure, which are the marginal distribution data and micro sample data for individuals and households. The datasets are described by CKAN resource IDs, and have to be represented as CSV files (in the future, HDF5 files will also be supported). For every input file, the version with the latest timestamp that is not more recent than the vintage date is selected.

The output data identifies the path to the CKAN dataset where the resources containing the result data will be created, the names of the resources, and the names of the data columns. The output data is written as a CSV file.

The job description is a JSON-formatted file that specifies which columns in the input dataset refer to geographical identifiers, categories (such as different combinations of income and age categories), and counts. The geographical identifiers and categories must be represented consistently in all datasets. Additionally, in case the microsample might contain attributes that are more detailed than categories in the marginal data, the mappings between attribute values and categories also need to be specified. These may be either mappings between categorical variables, or between a categorical variable and a range of an ordinal variable. The synthetic population generation procedure will use Iterative Proportional Fitting (IPF) in order to produce a contingency table matching the marginal distributions, and then use this table to sample households and individuals that comprise the synthetic population.

Currently, the execution of the job is not preceded by a separate stage where an execution plan is created. Instead, all errors are reported as they are encountered during the execution.

Synthetic populations produced by this step may be extended by networks generated by the network reconstruction tool, explained in the next subsection.

4.2 Network reconstruction tool Web GUI

A crucial ingredient in ABMs is represented by the modelling of interactions. For most of the social simulations, the main idea is that friendship relations influence the behaviour of single agents. Data about actual friendship (i.e. not based on social network platforms) are hard to collect: most of them are inferred by other measurements, like, for instance, mobile calls [14], [15]. In providing a network for the interactions among agents, we assume that similar agents influence each other, thus using the similarity among agents as a proxy for friendship.

The agents of CoeGSS pilots come from the generation of a synthetic population: as described in the previous paragraph, agents are defined by different attributes, whose global distributions satisfy the observed ones. The attributes can be of different types: integers, floats or categorical data. The similarity network needs to handle data of different kind and return a similarity value for each attribute, weighing them appropriately.

The problem lies in defining what is the similarity: in the literature, different proposals can be found, mostly basing their definition on the target of their analysis and thus lacking generalisation; moreover, most of the studies focus on just one data type (say, categorical data, continuous data, discrete data...), disregarding the possibility of heterogeneous datasets. In the network reconstruction tool, the definition of Lin [16] is considered and modified in order to overcome some drawbacks.

Indeed, Lin similarity has several positive features: it is general and unbiased, being based on Information Theory, and permits to analyse different data types. On the other hand, it implicitly considers all agent attributes as independent, an assumption that is rarely satisfied for real data.

In order to overcome possible correlations between attributes, in the present tool the Lin similarity per attribute is calculated, considering the data type, and then weighted using the Principal Component Analysis (PCA, [18]) results. In the PCA, the weights for the different attributes are calculated by considering the different contribution of the attributes to the scattering of the data. This contribution can be inferred from the components of the principal eigenvector of the covariance matrix.

As mentioned in [15], friendship relations are geographical distance dependent. In the tool, both exponential or $1/\text{distance}$ damping are examined. In the near future, other power law dampings are going to be considered. In the analysis so far, the produced networks show a strong hierarchical structure organised in dense clusters, as expected in social networks.

Essentially, the algorithm for the similarity network tool is divided in 3 steps: 1) the calculation of the weights of the similarities per attribute via the PCA; 2) the calculation of the similarity

for every pair of agents per attribute; 3) the calculation of the global value of similarity. In the last step, the different contributions of the similarity per attribute are weighted by the PCA.

In the current realisation, the input data is a .CSV file, as the one realised by MIDAS SPEW [19]. In the following months, the input file format will change to the HDF5 format as provided in the CoeGSS CKAN.

The user may decide to disregard some of the attributes of the synthetic population tool, considering the information they carry as unnecessary for the simulation. Since the approach is quite general, this selection is going to be part of the pre-processing of the network reconstruction tool: once data is suitably selected, the algorithm will be fed by the resulting dataset.

The code is implemented in Python, making use of mpi4py and h5py packages, i.e. Python wrappers respectively for MPI (Message Passing Interface) and the (parallelised) management of HDF5 files. The algorithm has just two parameters: the length scale at which the damping factor is equal to $\frac{1}{2}$ and a categorical entry controlling the damping function (exponential or $1/\text{distance}$, at the moment). In the next version, different power law exponents will be examined.

The output of the similarity network algorithm is going to be saved in the original input HDF5 file. A first HDF5 group is going to be created, representing the simulation step, then an HDF5 subgroup representing the processor used and then an HDF5 dataset for each attribute in the original dataset, representing the associated similarity calculated by the processor. An extra HDF5 dataset contains the global similarity, i.e. the one obtained from the weighted contribution of all similarities per attribute. Currently the results are saved in a new HDF5 file with the structure defined above.

Besides the modification mentioned above, we are examining different and more efficient ways of partitioning the set of agents for the parallelisation, due to the distance damping.

4.3 Visualisation tools

Within the project, the focus of the Visualisation Task is to develop and to provide remote and immersive visualisation services to consortium partners as well as to CoeGSS users. These services will be integrated into the CoeGSS portal regarding the defined workflow as described in D3.2 as well as the requirements described in D3.5. A main goal of these services is to provide access to high-performance, highly sophisticated visualisation integrated in a seamless manner in order to create an “Immersive Analytics Environment” for huge statistical and multidimensional datasets.

This subsection will describe the current status of development on interfaces to access datasets with respect to the proposed workflow definition as well as modules to read, process and visualise given datasets.

4.3.1 COVISE Portal Integration

With focus on visualisation of large datasets, three options to access visualisation are proposed according to the role of the user, given datasets and access to HPC resources.

I. remote session

After a simulation run on HPC resources, the simulation results usually are available via the user's workspace. Typically, the HPC infrastructure provides login, pre- and post-processing or visualisation nodes respectively with sophisticated visualisation hardware and high-bandwidth access to the workspaces. Available systems as well as hardware specifications on Hazel Hen at HLRS for instance are described in D3.5 chapter 4.3.2. These nodes provide dedicated VNC-sessions with OpenGL support to enable execution of COVISE/OpenCOVER for visualisation of simulation results. The link to a Virtual Network Computing session (VNC session) can be provided by the CoeGSS portal correspondent to the workflow progress.

An advantage of this option is, that processing as well as visualisation is running on the pre- and post-processing nodes within a HPC infrastructure and the user's local side just requires a VNC viewer. On the other side, a good network low-latency bandwidth as well as access to the HPC resources are needed.

II. hybrid session

COVISE/OpenCOVER is capable of running distributed visualisation sessions. Initially used for collaborative sessions, this feature can also be used to distribute different processes to multiple machines with specialised hardware configurations³. For example, a remote system with I/O and storage capabilities, which is used to read and process the data, and a local system with OpenGL rendering capabilities, which is used to render the data to the user's screen, can be interconnected within COVISE.

An advantage of this solution is, that storage and processing capabilities for these potentially huge datasets for visualisation are still provided by HPC resources, while visualisation can be run locally on a graphics workstation or a pc cluster when using a powerwall for example. This option enables high quality rendering dependent on the local rendering capabilities but still needs a good bandwidth.

III. local session

COVISE/OpenCOVER is available for Microsoft Windows™, Linux and Mac OS X. The install files can be downloaded via the HLRS website⁴, the source code is available via github⁵ and needs to be compiled for the local system. The advantages of this solution are that the visualisation session can run on a local machine stand-alone without the need of internet access nor access privileges to any HPC

³ <https://www.hlrs.de/solutions-services/service-portfolio/visualization/covise/features/>

⁴ <https://www.hlrs.de/solutions-services/service-portfolio/visualization/covise/support/>

⁵ <https://github.com/hlrs-vis/covise>

resources. This is a typical configuration used by CAVE setups or similar installations. Necessarily, datasets or subsets have to be downloaded to the local machine and processing as well as rendering needs to be performed by the local machine as well.

The three access options can be broken down into different steps needed, to access the data, to process the data for visualisation, to perform interactive visualisation and to report results.

a) interfacing data storage

Data needed for visualisation is either stored by the HPC storage system or provided by the user. Additionally, small to medium sized datasets can be hosted by the portal system directly. Typical data storage access is provided via ssh/scp, which can be linked to from the portal.

The procedure how to access the HPC workspaces depends on the HPC infrastructure of the HPC centre⁶. The type of services and procedures for getting access to the available system in the CoeGSS project are described in detail within D5.1 chapter 8.

Datasets have to be transferred or mounted to the compute system, which is running the associated reading instance of the COVISE net-file.

b) reading / processing data for visualisation

As mentioned the datasets for the visualisation need to be accessible by the associated reading instance of the COVISE net-file, while the interactive visualisation is performed by OpenCOVER, which is called by executing the same net-file.

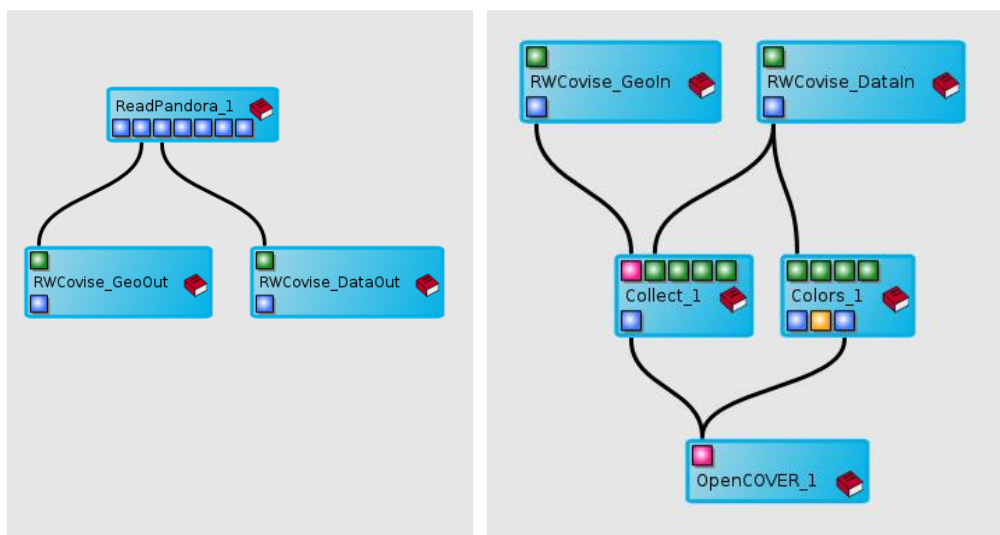


Figure 1. Writing into (l.) and reading from (r.) COVISE binary data (RWCovise)

Even though having all reading, processing and rendering modules in one net-file running on a single machine is the typical case, reading and processing may take a very long time depending on input formats, size and performance of the local machine as

⁶ https://wickie.hlr.de/platforms/index.php/Cray_XC40

well as the reader implementation. To prevent running interactive jobs for a long time or just to prevent longer waiting periods before the visualisation starts, the COVISE-module `rwcovise` can be used, to read data into a `covise` binary file first and read from the COVISE binary file with further processing modules later on. Hence, the net-file is being split into a non-interactive processing part and an interactive visualisation part. Reading COVISE binary files can be more efficient in many ways, for example, if only a subset of the data or parameters are of interest or data has to be parsed from ASCII-files. Additionally, the COVISE net-file, which runs reading as well as processing modules can be run as a batch job on I/O nodes for instance. A net-file (Figure 1 I.) can be run as a batch job as follows.

```
#covise example1.net --nogui -q
```

The `--nogui` parameter prevents COVISE from starting a graphical user interface showing the net-file and pop-up windows, thus no display is needed. The `-q` parameter terminates COVISE as soon as the net-file is processed.

However, one drawback with this procedure is that the user has to rerun the reading and processing net-file every time if he wants to change the subset of data or he wants to choose different parameters for the visualisation. Of course, the user can run it multiple times in advance, if he wants to have multiple views or subsets like a data set for every country in Europe instead of a complete dataset of Europe for example. Incidentally, multiple COVISE data files can be loaded into one interactive visualisation session at the time. The description and example files of the most important COVISE modules like Converter, Filter and I/O-modules can be found within the COVISE online documentation⁷.

c) *interactive visualisation and user interaction*

As mentioned in the beginning of this section, there are three options to run visualisation on CoeGSS datasets. In addition to these three options, the processing can be partitioned into reading/pre-processing and visualisation. With pre-processing data for the visualisation done by a separate possibly non-interactive net-file, an interactive visualisation can be started with minimal time needed to read/prepare the data. As mentioned before, multiple datasets can be loaded for visualisation at the same time. Here, the user has to find the trade-off between flexibility, pre-processing and visualisation in the same session, and minimal time needed, batch pre-processing and visualisation in different sessions.

⁷ <https://fs.hlr.de/projects/covise/doc/html/index.html>

While installing COVISE/OpenCOVER to a local machine is a common task for experienced users, setting up a remote session or even a distributed session is a bit more challenging. Setting up a distributed COVISE session requires access to all involved machines. The COVISE session is controlled by the user from the MapEditor on the local workstation.

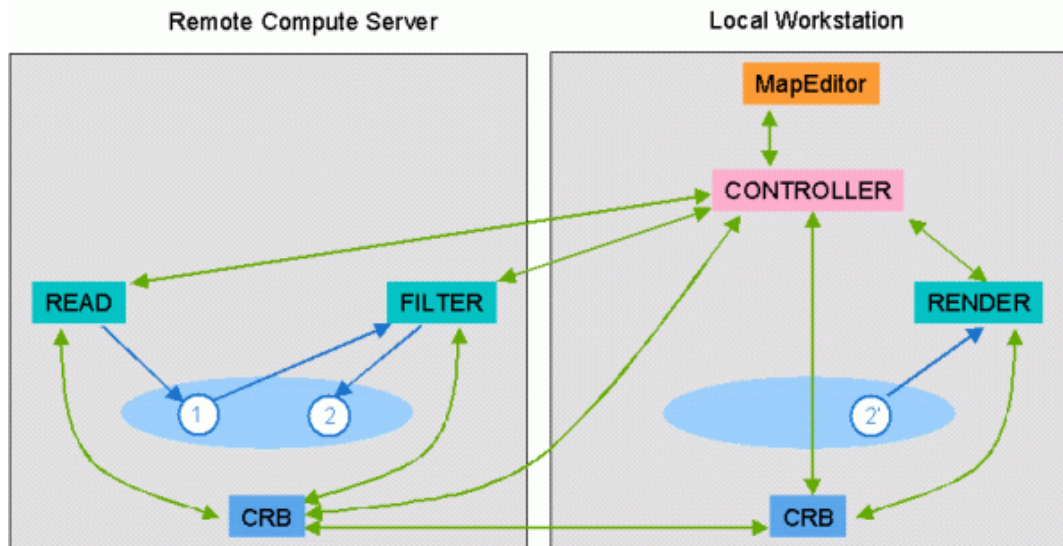


Figure 2 Distributed COVISE Session

Figure 2 shows the elements of an example for distributed working in COVISE. The application consists of three modules: a module which reads in data (READ) a module which extracts a special feature (FILTER) and a module which displays the extracted data (RENDER). As the filter module consumes much CPU time and memory, it will be started on a remote compute server, as well the Reader because the data to be read in is on the remote machine. The first process started by COVISE is the Controller which in turn starts the user interface process MapEditor and the data management process the COVISE Request Broker (CRB). As soon as another host is included in the session, a CRB is started on that computer. The read module is started on the local workstation, the filter module on the remote computer and the renderer on the local workstation. The green arrows between the processes Controller, MapEditor, CRB and the modules indicate TCP sockets, the blue arrows indicate shared memory access.

When the module net-file is executed, the Controller sends a start message to the remote read module. The read module reads in the data file and creates a COVISE data object (1) in shared memory and after processing tells the controller that the module has finished. The controller informs the filter module on the remote computer to start. The filter module asks its data management process (CRB) for the data object (1). The filter module now reads that data object, computes something and puts the data object (2) into shared memory. It then tells the controller that it has finished. The controller informs the renderer module to start. The renderer asks the CRB for object (2) and as this object is not available on the local workstations the CRB transfers it from

the compute server into the shared memory of the local workstation (2'). Now the renderer can access this object and display the data.

Important to note here is, that memory intensive objects (1) only have to be stored in memory on the remote compute server, while the filtered data (2) is exchanged with the local workstation. Setup of distributed sessions is documented in detail within the COVISE online documentation⁸.

Downloading, installing and using COVISE/OpenCOVER for post-processing and visualisation on a local machine is possible anytime, installer files are available via the support websites⁹ or via the github repository¹⁰. If the user has access to HLRS Hazel Hen, COVISE/OpenCOVER can be used within an interactive VNC session via the login/pre- and post-processing nodes as mentioned above.

d) reporting

Users have different possibilities to document results or specific views as well as to write output files accordingly. To document specific views via screenshots of the rendering currently observed, the OpenCOVER plugin PPTAddIn as already presented in D3.5 can be used. This plugin enables users to define and to link screenshots instantly into a Microsoft PowerPoint presentation for example, to track progress of an investigation or data analysis.

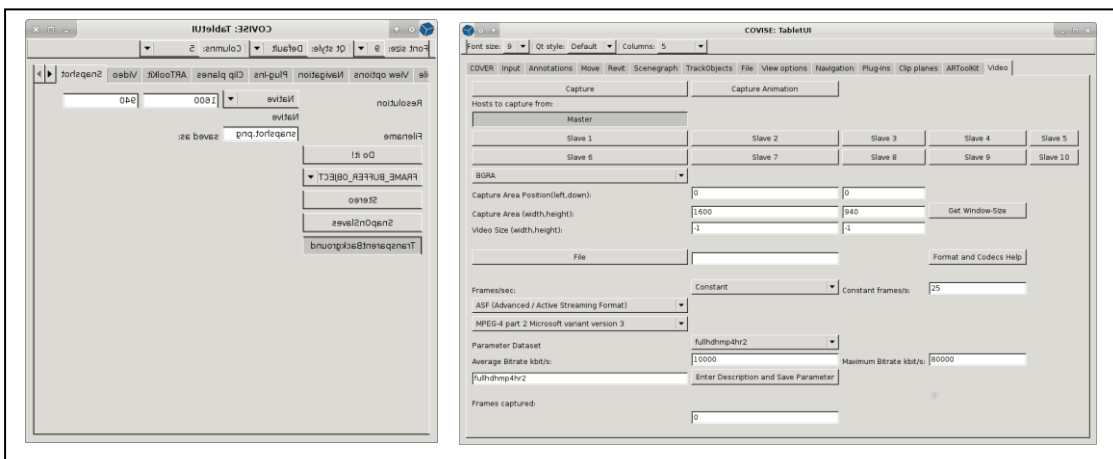


Figure 3. OpenCOVER SnapShot and Video Plugin

Furthermore, the OpenCOVER plugin can be used to snap screenshots from the current view or to record videos about the user's view and interaction in the scene or the data he is exploring (Figure 3).

To write/export data after processing for visualisation, various I/O-modules can be used to write data into specific or user defined formats. A list of I/O-modules including documentation can be found within within the COVISE online documentation¹¹.

⁸ <https://fs.hlrs.de/projects/covise/doc/html/usersguide/collab/collab.html>

⁹ <https://www.hlrs.de/solutions-services/service-portfolio/visualization/covise/support/>

¹⁰ <https://github.com/hlrs-vis/covise>

¹¹ <https://fs.hlrs.de/projects/covise/doc/html/usersguide/collab/collab.html>

4.3.2 Examples from the pilots

In the following chapter, an overview of visualisation examples on datasets from the pilots is given. These are not meant to demonstrate specific discoveries or characteristics within the given datasets themselves, but to present possible use cases of visualisation tasks.

Pilot 1: Health Habits – Smoker Study

In COVISE it is possible to read selected or all time steps of a simulation run, which are usually visualised as an animation. Another possible visualisation is demonstrated in **example 1** (Figure 4, Figure 5), in which each time step is shown by a separate grid. This enables the user for example to investigate the development of hotspots in the data visually without the need of going forward and backwards on a time line.

This is being done within the StackSlices-module, instead of mapping the read time steps into a time sequence, the time steps are read into different slices, which can then be oriented along the x-axis for instance. Specific time steps, which shall be compared by the user, may be selected within the GetSubset-module.

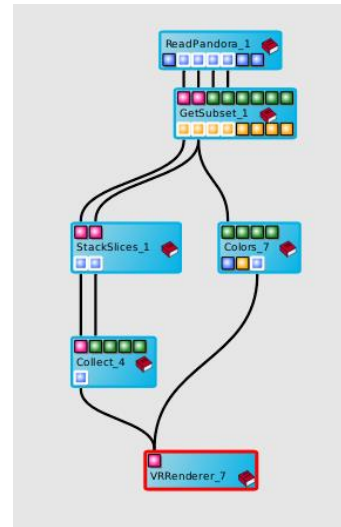


Figure 4. Covise net-file (example 1)

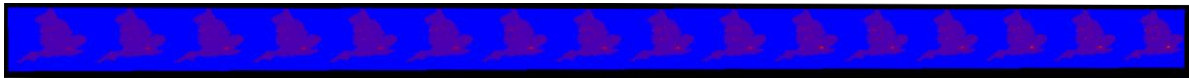


Figure 5. Opencover output – visualisation of a time sequence (example 1)

As shown in **example 2**, the same approach can be used to investigate the development of various parameters next to each other (**Figure 12**). In this case all time steps of the parameters were loaded in separate animated data grids. With running the animation of the time steps, the user can investigate and compare the development of these parameters in parallel. Spatial separation is done by using the Transform-module, which can be used to place the grid files

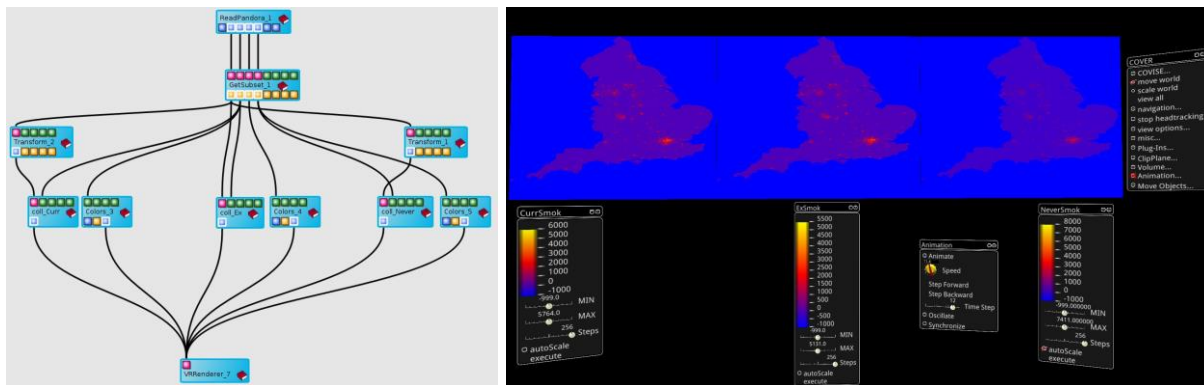


Figure 6. Visualisation of specific parameters over time (example 2)

in any 3D-position the user prefers. The scales of all three parameters are shown, which can also be changed to different colour mappings.

In **example 3 (Figure 13)**, the module DisplayUsg is being used. This module enables mapping specific parameters from the dataset to a free selectable axis in space. In this case, one parameter is mapped to the height of the map, while another parameter is mapped to the colour of the map. This way the user can observe differences in the data represented in different modalities of the visualisation. Also shown in this example, the processing of the data is distributed to two separate machines. Indicated by different colours of modules in the

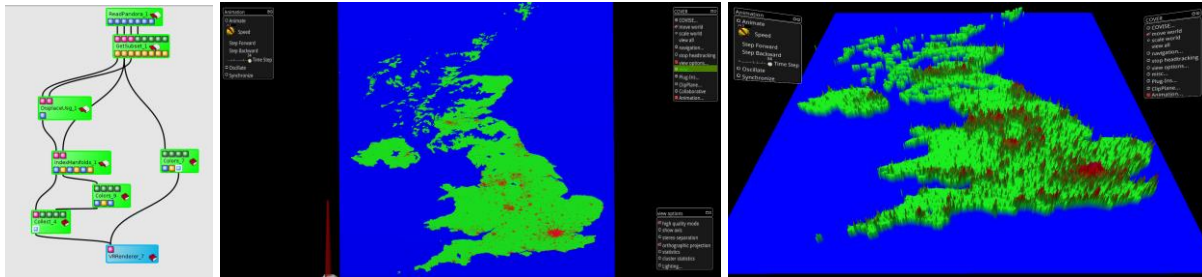


Figure 7. Height displacement of a specific parameter (example 3)

COVISE net-file, reading and processing is done by a server system, and the visualisation is done by a laptop. This setup was chosen because the laptop has not enough memory to load and process the datasets but has a capable graphics card to do the visualisation.

If the user wants to compare the height map of two different parameters, these two different parameters may be mapped to different directions of an axis for instance. **Example 4** shows two different parameters, one mapped to the -y-axis and the other one to the +y-axis in orthographic projection to prevent distortion of the data by perspective projection (Figure 8).

Pilot 2: Green Growth – Green Cars

Height maps have also been used in Pilot 2 – Green Growth to demonstrate using different modalities on different parameters within the dataset shown in **example 5** (Figure 10, Figure 9). A specific time step of the data is used to perform height displacement of the data grid.

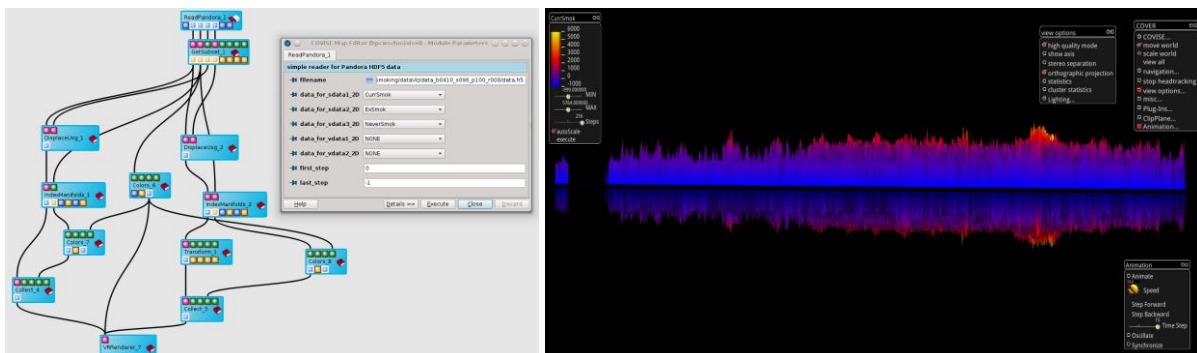


Figure 8. Comparing height maps (example 4)

The user may also use clipping of the data or the height map respectively to investigate the height displacement in specific regions of the data.

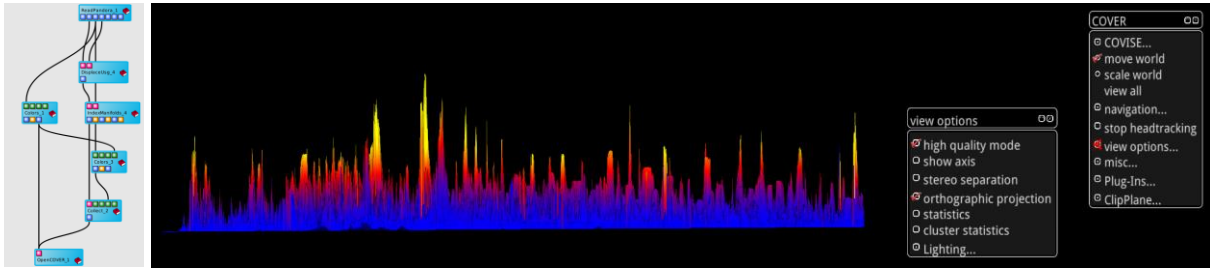


Figure 10. Height displacement in orthographic projection (example 5)

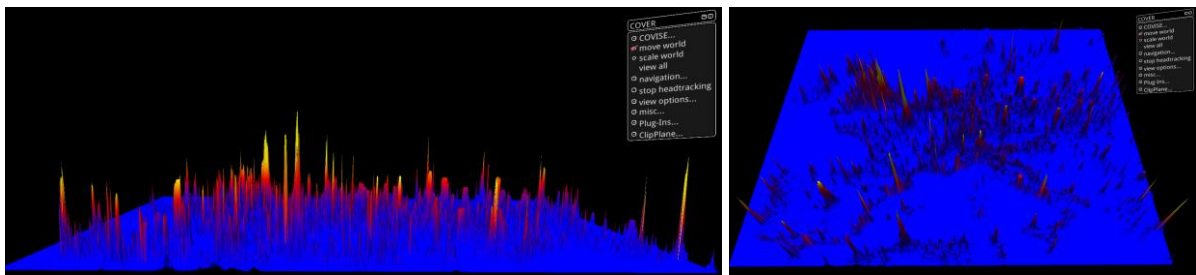


Figure 9. Height displacement in perspective projection (example 5)

The COVISE module StackSlice is usually used for medical datasets like CT-Scans for instance, to stack a set of 2-dimensional images to get a 3D volume to perform volume rendering. **Example 6** presents a use case, in which this module was used to stack 2-dimensional slices, with each slice representing the value at a specific time of a parameter (Figure 11). In this case the parameter is a density value. Using clipping or the transfer function editor, the user can observe spreading characteristics of the selected parameter.

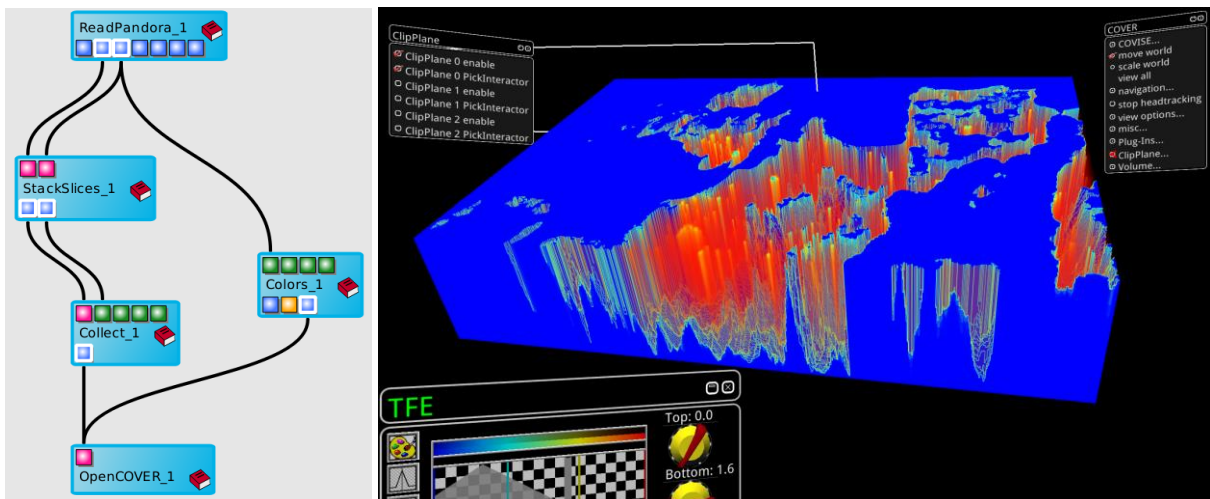





























Figure 11. Volume rendering of a time sequence (example 6)

4.3.3 List of requirements / status

The following table lists given requirements addressed by reports or deliverables so far. Even though not all requirements can be resolved within the project due to needed effort or missing dependencies respectively, future development of COVISE/OpenCOVER usually focuses different aspects depending on users' requests. Questions or requests may be addressed to the COVISE mailing list¹² anytime.

category	description	reference	status
METHODS			
	description	reference	
	visualise the results of SI simulations	D4.1 (4.4)	
	interact with simulation results in real time	D4.1 (4.4)	
	visualization of full-blown runs, another tool that allows the visualization of GIS data and time-series of statistical figures	D4.1 (6.7)	
	compare multiple runs of the model	D4.1 (6.7)	
	brush subsets of data points	D4.1 (6.7)	
	two dimensional maps of cities	D4.1 (7.5)	
	if possible, unfolding different features (population, traffic, prices, pollution, etc.).	D4.1 (7.5)	
	analysing and interpreting the resulting data (general req)	D3.1 (4.2)	
	methods to process huge and varying volumes of unstructured data	D3.1 (4.2)	
	methods for additional data management.	D3.1 (4.2)	
	can handle incomplete information	D3.1 (4.2)	
	remote visualisation	D3.1 (4.2)	
	raw mode visualisation	D4.2 (4.1)	
	visualisation of geo-referenced data on a map	D4.2 (4.2)	
	compute different aggregations based on shape-files (e.g. regional and country level data)	D4.2 (4.2)	
	switch between cases	D4.2 (4.2)	
DATA INTERFACE			
	description	reference	
	HDF5 / Pandora Format		
	geo data is gridded on a 3432x8640 raster and encoded as geotiff	D4.1 (6.5)	
	GSS synthetic population simulations	D3.1 (4.2)	
	Structured and unstructured data	D3.1 (4.2)	
	Regular and irregular patterns (lists, matrices, graphs)	D3.1 (4.2)	
	Read CSV	D3.1 (4.2)	
	I/O modules / general expandability	D3.1 (4.2)	
	CKAN interface	D1.3 (5.2)	
	access CKAN data directly	D3.5 (4.3)	
	access CKAN data by reference	D3.5 (4.3)	
	automation of defined processing	D3.5 (4.3)	

¹² <https://listserv.uni-stuttgart.de/mailman/listinfo/covise-users>










	process of data treatment must be tracked	D4.2 (4.1)	
	automated and generic extraction from a given file and aggregate it according to a given specification	D4.2 (4.2)	
	support GIS raster data	D4.2 (7.4)	
	import HDF5 tables	D4.2 (7.4)	
TOOL INTERFACE	description	reference	
	Pandora	D4.1 (5.6)	
	GLEAMviz simulator tool		
	ggobi (http://www.ggobi.org/)		
	CoSMo modelling software		
	Hadoop	D3.1 (3.3)	
	Apache Cassandra	D3.1 (3.3)	
	R Project		
	integrated versioning system for data sets (would be useful)		
DATASIZE	description	reference	
	large populations of up to a hundred million individuals	D4.1 (5.6)	
	first tests using a simulation with about 150.000 agents and 100 time steps (the epidemic example of Pandora)	D4.1 (6.7)	
	support large number of agents (billions) and related data	D4.2 (7.4)	
ACCESS	description	reference	
	access visualization tools on HPC systems (resource management)	D4.1 (4.3)	
	web based access to the visualisation	D3.1 (4.2)	
	data sets can be handled as private	D4.2 (4.1)	
	hiding parallel MPI code completely from the user	D4.2 (7.1)	
OS SUPPORT	description	reference	
	SUSE Linux SLES11	D4.1 (7.5)	
	Windows 7 64 bit		
	Windows 8.1 64 bit	D4.1 (7.5)	
	Windows 10 64 bit		
	Debian 7 (wheezy) 64 bit	D4.1 (7.5)	
	Debian 8 (jessie) 64 bit	D4.1 (7.5)	
	Ubuntu 14.04 LTS 64 bit	D4.1 (7.5)	
	OS X 10.10 64 bit	D4.1 (7.5)	
Hardware	description	reference	
	3D virtual environments like CAVEs or a Powerwall	D3.1 (4.2)	
	2D Desktop	D3.1 (4.2)	

Table 1. Visualization requirements

The status is given as follows (Table 2).






	fixed or solved
	partially solved, in development, under discussion or further tests needed
	open issue or request
	won't fix because outdated or out of focus
	Needs further refinement

Table 2. Legend for table of visualization requirements.

4.4 CKAN data converter

The input data are imported from outside databases to the CoeGSS data management system (based on the CKAN server) by the CKAN harvester, implemented as a CKAN plugin. During the importing procedure data from outside sources are converted to:

- Database tabular format (easy to access and explore directly from the portal)
- HDF5 format

4.4.1 HDF5 conversion procedure

The conversion procedure is implemented in the CKAN DataPusher plugin. The plugin is a standalone web service that automatically downloads any CSV, TXT or XLS (Excel) data files from a CKAN site's resources when they are added to the CKAN site, parses them to pull out the actual data, then uses the DataStore API to push the data into the CKAN site's DataStore.

We have implemented new methods to convert data from the specific input files to the HDF5 output format. We use a h5py Python library¹³. The newly implemented methods are saved in a new file called `hdf5convert.py` in the DataPusher directory.

The HDF5 conversion procedure can be run in two ways:

- in one step - when the DataPusher plugin reads input files,
- in many steps - when the DataPusher plugin parses input records.

We have modified the DataPusher plugin source code in `jobs.py` file. After a successful conversion to the HDF5 format the new file named `[ORIGINAL_FILE_WITH_EXT].H5` is uploaded to the same dataset as the input file CSV/TXT/XLS(X).

¹³ <http://www.h5py.org/>



Figure 12. The original CSV and HDF5 file in CKAN dataset.

4.4.1.1 Conversion from CSV, TXT

The first row in the CSV or TXT files is a header with titles of columns, and the other rows are values. Since information about data types is not available, all values in the HDF5 output file are going to be saved as `strings`. The lengths of characters are recognized in individual columns and then a HDF5 file is created and filled in by data. The rows in which the number of columns is different from the number of columns in the header are omitted.

By default the table name in the output HDF5 file is “data”.

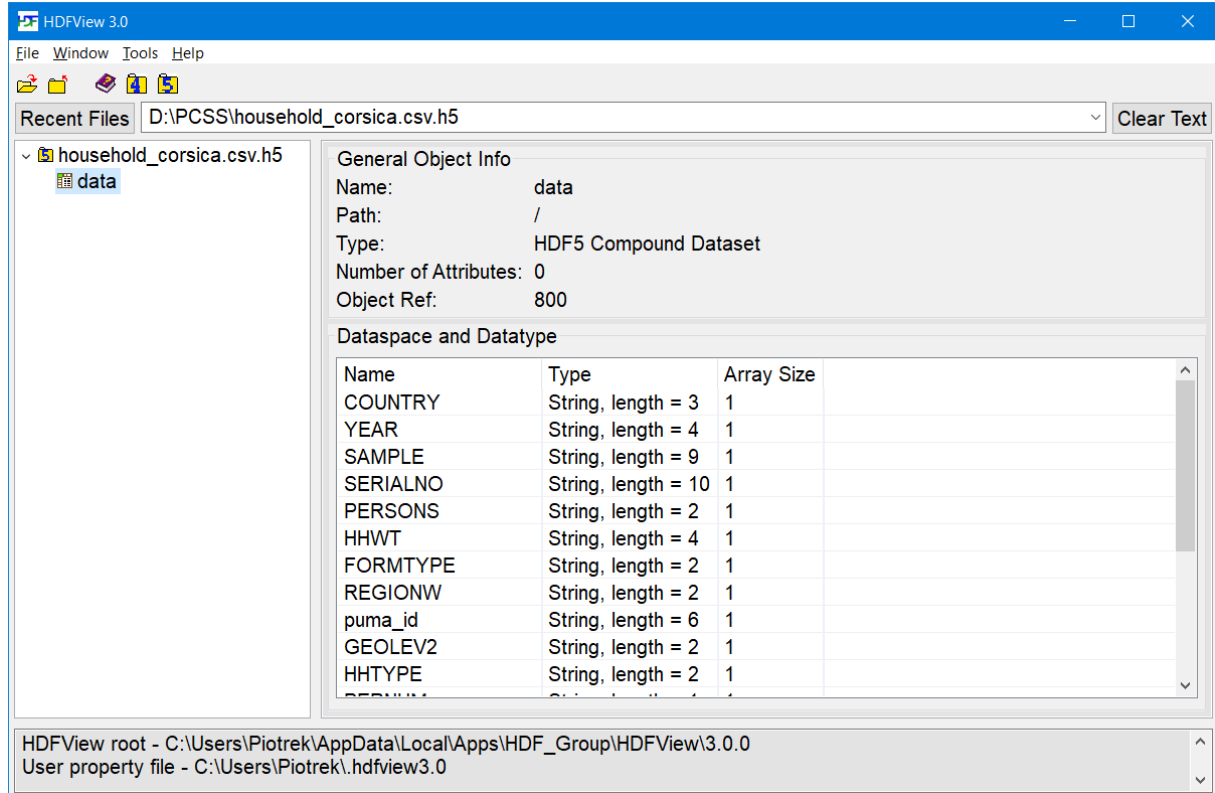


Figure 13. The data converted from the CSV file.

4.4.1.2 Conversion from XLSX

Similar to a CSV format, the first row in the XLS(X) file is a header with titles of columns, and the remaining rows are values. Again information about data types are either not provided or could be provided wrongly, so all values in the HDF5 output file are transformed to strings. The length of characters in individual columns is recognized and then an HDF5 file is created and filled in by data.

This implementation is supported by a python library xlrd <https://pypi.python.org/pypi/xlrd>. By default the table name in the output HDF5 file is "data". Only the first sheet from XLS(X) files is converted.

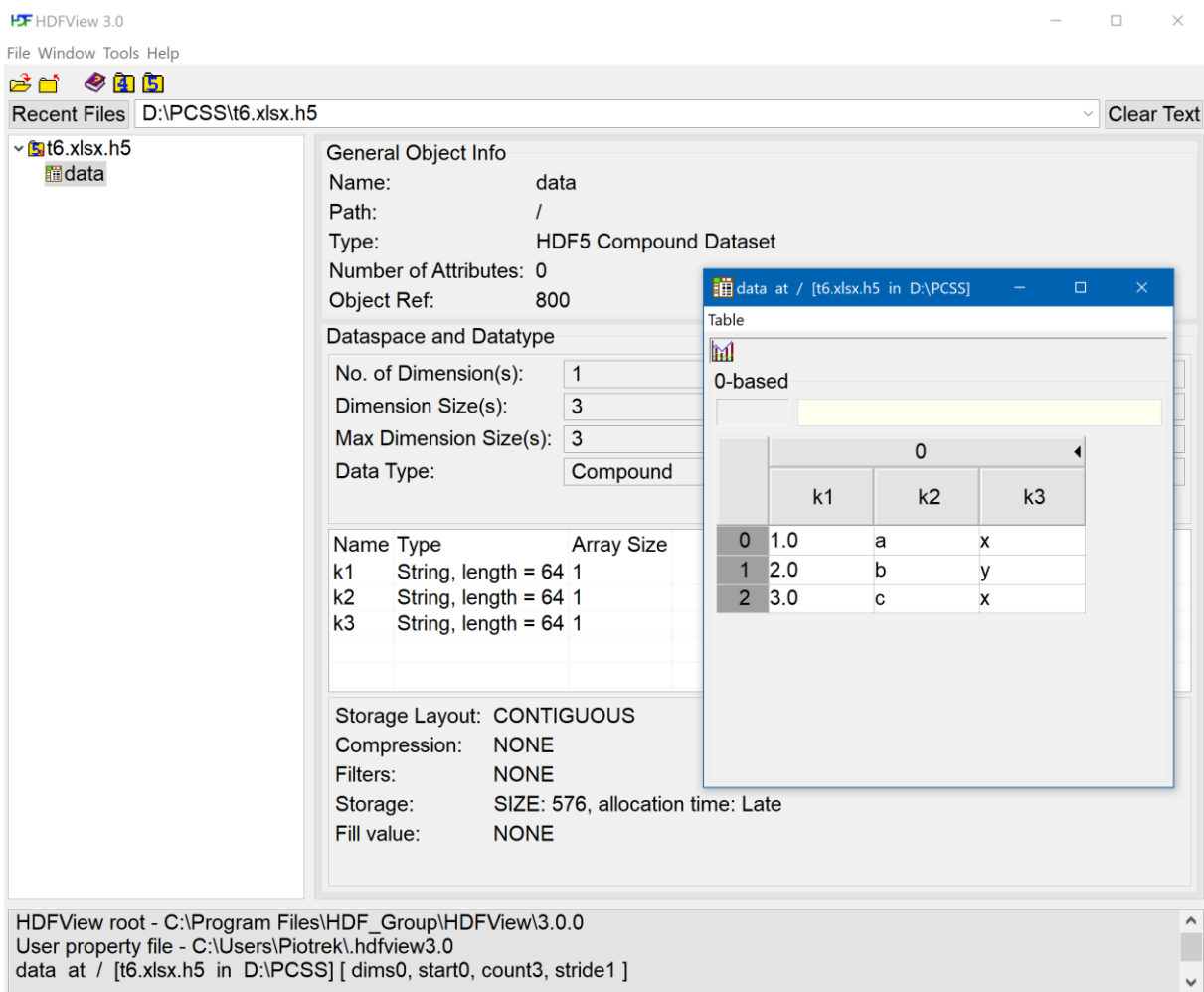


Figure 14. The data converted from the CSV file.

4.4.1.3 Performance tests

Extensive performance tests were conducted for CSV files of different sizes. Files were imported from the MIDAS database¹⁴.

FILE SIZE / LINES	CONVERSION TO HDF5	FILE NAME
8 KB / 84 Lines	0.6 s	iceland_admin.csv
20 MB / 167553 Lines	1m 40s	household_vorarlberg.csv
74 MB / 640261 Lines	6m 55s	household_sardegna.csv
115 MB / 765022 Lines	7m 16s	household_theautonomousrepublicofcrimea.csv
199 MB / 1336855 Lines	11m 15s	people_westberlin.csv
255 MB / 1751555 Lines	16m 26s	people_sardegna.csv

Table 3. Conversion time to hdf5 output format

As was expected, the conversion time grows nearly linearly with the number of rows to be processed.

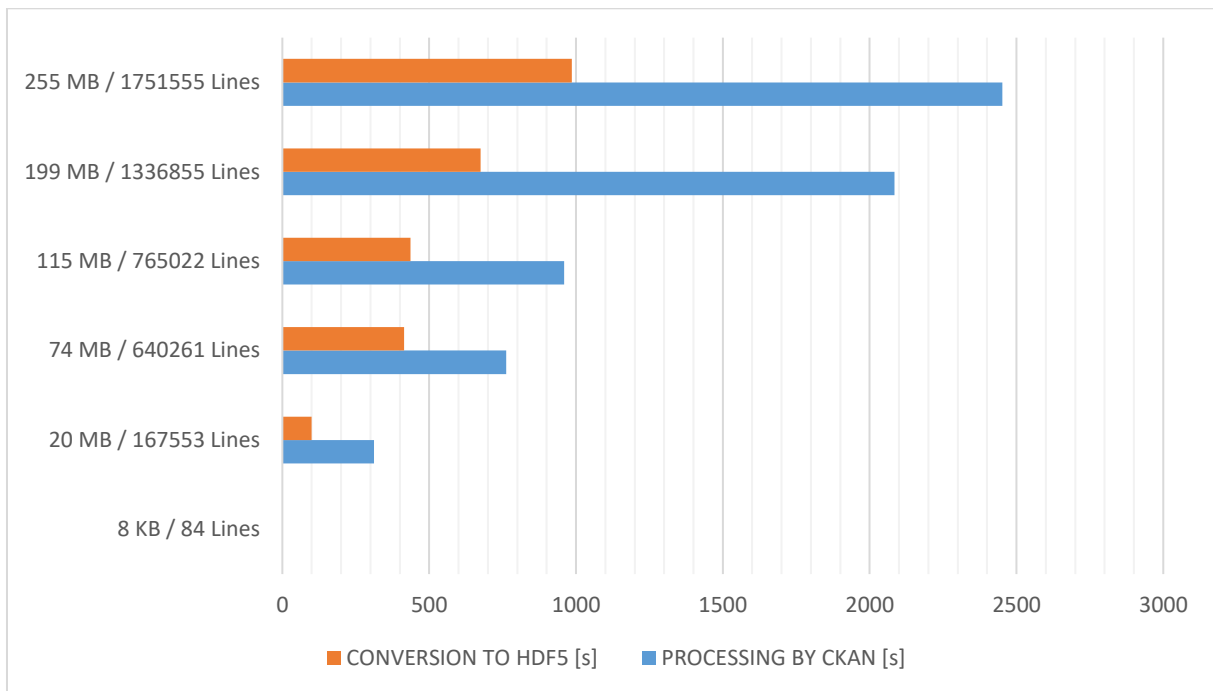


Figure 15. Conversion time to HDF5 format and processing time of the CKAN database vs file size.

¹⁴ MIDAS <http://data.olympus.psc.edu/>

As the number of rows grows, the conversion time to HDF5 format will be faster than the processing time to the database tabular format. Results for XLS(X) and CSV/TXT input files are very similar and depend only on the number of rows in the input files.

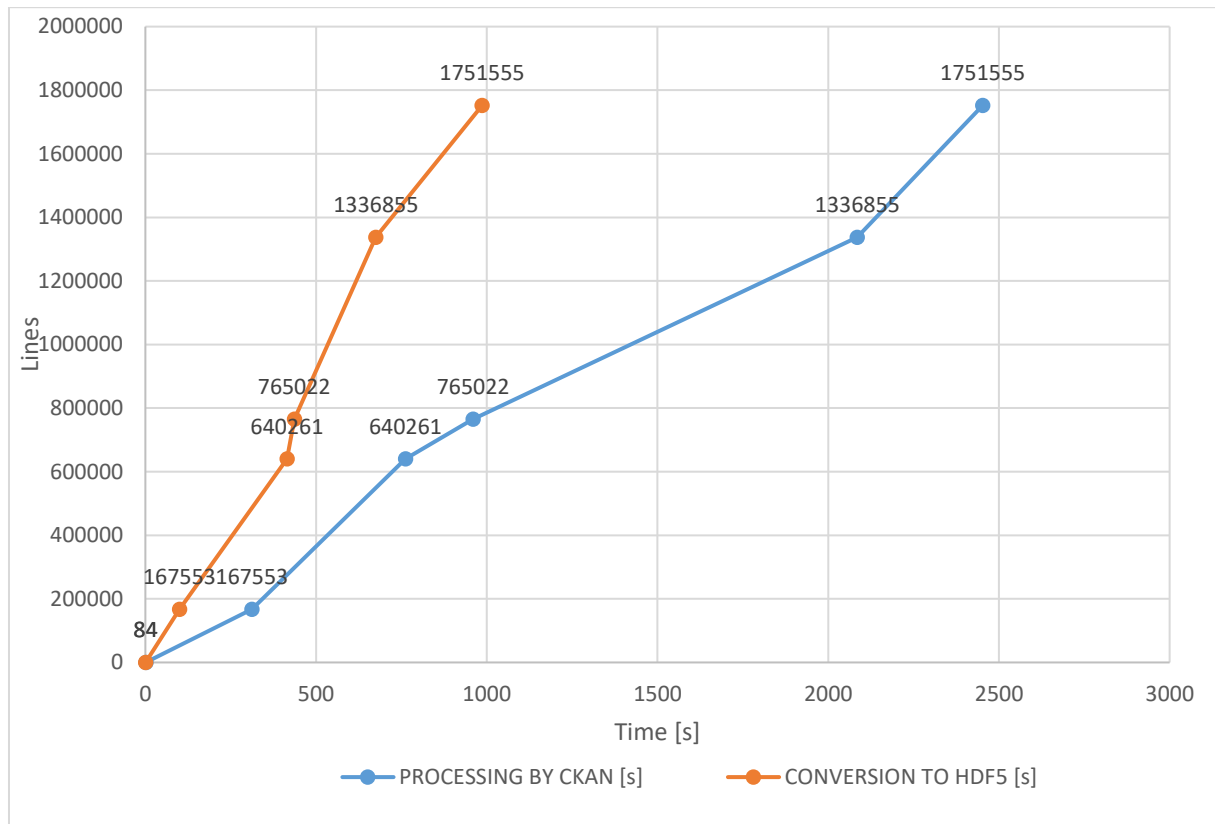


Figure 16. Conversion time to HDF5 format and processing time of the CKAN database vs number of lines

The first “one step” method of conversion to HDF5 format (SUM PROC + CONV HDF5) is slower for input files less than 700 000 rows.

When the input file has more than 700 000 rows, the summary time of processing to database and conversion to HDF5 format is shorter than the execution time of the second method (PROC & CONV TO HDF5).

The reasons of this fact are:

- conversion from a CKAN JSON input row to a Python table data,
- detection of string type value and if necessary encoding to UTF-8,
- resize time of the HDF5 dataset.

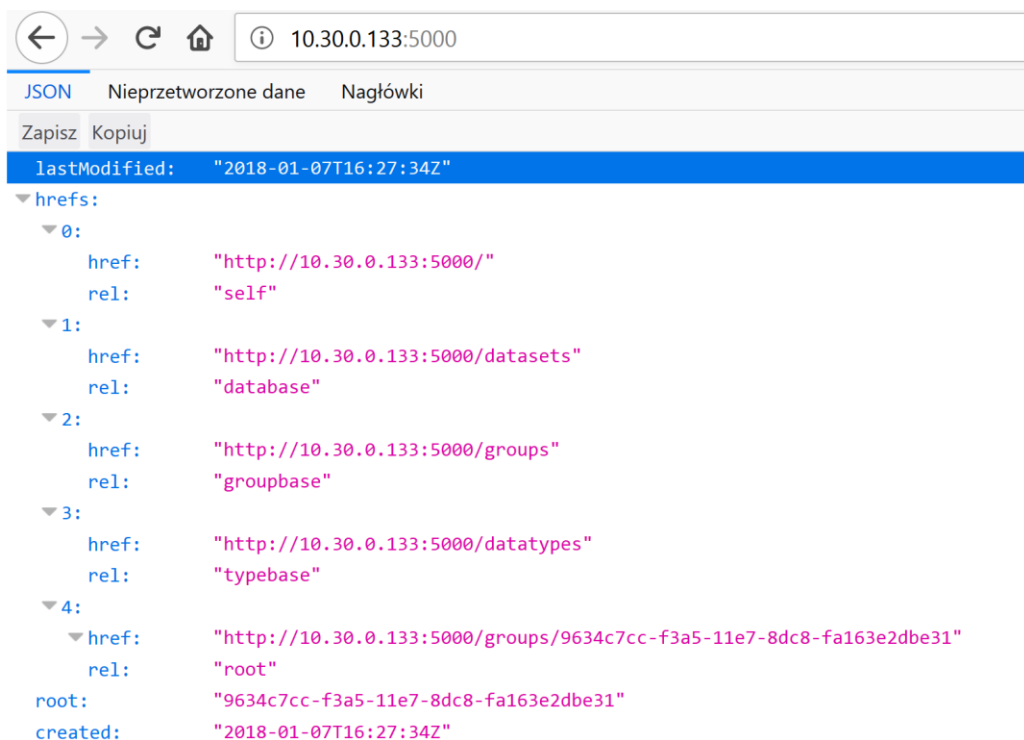
In the second method, the HDF5 output files are much larger than input CSV files and HDF5 files from the first method. As the row count and max string length of values in individual columns are not known, a length of 32 characters is assumed.

4.4.2 Exploring HDF5 files from the portal

There is a solution for browsing HDF5 files that can be integrated with the CKAN portal HDF Server¹⁵. It is a Python-based web service that can be used to send and receive HDF5 data using an HTTP-based REST interface. HDF Server supports CRUD (create, read, update, delete) operations on the full spectrum of HDF5 objects including: groups, links, datasets, attributes, and committed data types. As a REST service a variety of clients can be developed in JavaScript, Python, C, and other common languages.

The HDF Server extends the HDF5 data model to efficiently store large data objects (e.g. up to multi-TB data arrays) and access them over the web using a RESTful API. As datasets get larger and larger, it becomes impractical to download files to access data. Using HDF Server, data can be kept in one central location and content vended via well-defined URIs. This enables exploration and analysis of the data while minimizing the number of bytes that need to be transmitted over the network.

The h5serv was designed as a Web API, not a Web UI (though it is a great platform for building a Web UI). Currently, the server always returns responses as JSON formatted text.



```
lastModified: "2018-01-07T16:27:34Z"
hrefs:
  0:
    href: "http://10.30.0.133:5000/"
    rel: "self"
  1:
    href: "http://10.30.0.133:5000/datasets"
    rel: "database"
  2:
    href: "http://10.30.0.133:5000/groups"
    rel: "groupbase"
  3:
    href: "http://10.30.0.133:5000/datatypes"
    rel: "typebase"
  4:
    href: "http://10.30.0.133:5000/groups/9634c7cc-f3a5-11e7-8dc8-fa163e2dbe31"
    rel: "root"
root: "9634c7cc-f3a5-11e7-8dc8-fa163e2dbe31"
created: "2018-01-07T16:27:34Z"
```

Figure 17. The default output from h5serv.

¹⁵ <https://www.hdfgroup.org/2015/04/hdf5-for-the-web-hdf-server/>

Unfortunately, the HDF5 Group does not support the HDF Server for now. The WEB UI is not available. The new project of web access to a HDF5 data is a cloud solution, which is not addressed by our project.

5 Software outlook

This section highlights ideas and plans regarding integration of the software components into further releases of the portal. The current status of development of ABMS tools (Pandora, AMOS) is presented with special emphasis on interfacing aspects. Next, an approach of coupling the data analytics tools with the portal is described.

5.1 ABMS interface

5.1.1 Simulation workflow and lifecycle of the simulation experiment

When using the portal for ABMS, interaction between the end user and the portal consists of three steps – setting up the simulation experiment, launching the experiment, and browsing the simulation results.

In the first step, the user applies a Web-based *setup assistant* to specify the simulation experiment. It allows to choose the simulation framework, upload model sources, choose the target hardware, define compilation options, select input/output files, and specify model parameters. For further details on the simulation experiment wizard, we refer to the next subsection.

As soon as the simulation experiment is prepared, it can be launched. The lifecycle of the launched experiment consists of the following sequence of actions:

- *creation of the workspace*

This action implies loading an appropriate environment and deploying the model and input data on the target platform. The portal's software opens a session for simulation and loads an environment required for proper compilation and linkage. Next, it downloads model sources along with requested input files from CKAN and puts them into an empty folder prepared for the simulation experiment. Finally, it generates model configs if the selected ABMS framework requires some.

- *configuration, compilation, and linkage of the model*

After preparing the workspace, the deployed model is configured, compiled, and linked against the target framework libraries into an ABMS executable. If the model sources lack configuration script (scons-script in case of Pandora or cmake-script in case of Amos), the default one is generated. If this action fails, the user will be notified about an error via email with attached configuration, compilation, and linkage logs.

- *job submission*

Depending on the job scheduling software of the target platform, either a Slurm or PBS job script is generated and submitted. This job script may also include some command line arguments for the ABMS executable if the user has specified those in the simulation experiment setup.

- *uploading simulation results to CKAN and notifying the user*

Once the job is finished, the simulation output is uploaded to CKAN. Afterwards, the user is notified about job completion via email which contains links to the simulation results and the profile of the experiment.

All these actions are performed automatically.

In response to the job completion notification, the user can browse the simulation results. If further model runs are required, the user may update model parameters and/or target the platform and relaunch the experiment.

5.1.2 Input/output parameters setup

The ABMS experiment setup is usually performed with aid of a corresponding Web-based setup assistant included into the portal. In this subsection, we present a general interface of the setup assistant and discuss interfaces of ABMS frameworks chosen to be introduced into the portal.

Setup assistant

The Web-based setup assistant helps to specify details of the simulation experiment. In order to accomplish this task, it guides the user through a sequence of 4 dialogue boxes.

In the first dialogue, the user is asked to specify the model, the ABMS framework, and the third-party libraries (if model needs the latter). The model sources should be previously uploaded to the CKAN server.

In the second dialogue, the user specifies locations for the input and output files and defines values of the simulation parameters. The input files should be previously uploaded to the CKAN server in a similar way to the model sources. The simulation parameters usually include a number of simulation steps, a serialisation step, etc. In addition, the user can introduce and assign values to the model-specific parameters if the model depends on some attributes missing in the list of general parameters.

In the third dialogue, the user selects a compiler and (if needed) compilation options.

Finally, in the fourth dialogue, the user specifies hardware for the simulation. Namely, he/she chooses the target platform and defines job parameters (such as required number of cores, name of the job queue, etc.). In the combo-box, the setup assistant lists only those target platforms of the HPC centres, which have a software defined in the first and the third dialogues.

Pandora interface

As reported in D4.4, different existing ABM frameworks with HPC support were evaluated, and Pandora has been chosen as one framework that should be part of CoeGSS tool chain. The global model of the Green Growth pilot and the compartmental model of the Health Habits pilot are implemented in Pandora (see e.g. D4.5).

Beside the job submission and monitoring, the interface between Pandora simulations and other tools are completely file-based.

The configuration of a simulation is defined in an XML file, per default Pandora tries to read a file called `config.xml` from the same directory as the simulation binary. This file must contain the following tags and attributes:

- `<output>`: The output tag has two attributes that define the file/directory name of the generated output, the `resultFile` is described below, and log files will be written to `logsDir`
- `<numSteps>`: `value` determines the number of steps that the simulation should run. Every `serializeResolution` steps the state of the simulation is written to the output files.
- `<size>`: The `width` and `height` attributes defines the spatial size of the simulation

Beside those mandatory attributes, the user can add arbitrary tags and attributes to the xml-file and access them easily using the `getParamStr`, `getParamFloat` ... functions. The standard defined for the CoeGSS tool chain is to use the tag `params` for attributes which can be different for each simulation run, and can be set by other tools e.g. for the model calibration.

Pandora does not come with built-in support for synthetic populations, or even more general with built-in support for reading other files than the configuration file. PSNC has written an Amos plugin that allows to read HDF5 tables in general, and synthetic populations in particular. It is planned to adapt this plugin to the Pandora framework.

The output files of Pandora are described extensively by the Pandora documentation distributed along with the Pandora package. The raster files can be easily read by tools that support the HDF5 file format, like e.g. in R, using the `h5read` function of the `rhdf5` package. For the agent files a R function was written, that allows to aggregate the values of a registered attribute for each step to create a time series for this attribute.

AMOS interface

As mentioned in the previous paragraph, Pandora covers basic needs related to the simulation of agent-based models with GIS inputs, but it has a limited support of synthetic populations. Moreover, it lacks any internal tooling for doing simulations with synthetic networks. And while the HDF5 I/O plugin developed by PSNC should introduce support of synthetic population input in Pandora, it cannot resolve the issue with synthetic networks. At the same time, synthetic networks play an important role in modern GSS research. In order to cover this gap, the portal requires an alternative framework which supports synthetic networks. As one of the possible solutions, the Amos framework can be used.

Amos is an agent-based modelling toolkit intended for high-performance distributed computing platforms. This project aims to equip global systems scientists with a user-friendly

programming environment for constructing models without knowledge of low-level details of parallel computing. Unlike Pandora, Amos allows agent interactions not only via spatial relationships, but also via synthetic networks representing social relationships.

Amos is implemented in modern C++ using MPI for parallel communications. It is designed trying to minimize the dependencies on third-party libraries in the core of the framework. Apart from MPI, the recent version of the Amos core makes use of the Boost libraries only. At the same time, the Amos design presumes that the core functionality can be easily extended via plugins. Such plugins intend primarily to introduce new I/O formats and workload distribution approaches, as well as to access third-party graph libraries. The sources of the framework are configured and assembled by means of the CMake build system. The framework uses the Google Test library for unit testing.

Amos inherits major architectural solutions from the Repast HPC framework [8]. In particular, following Repast HPC, Amos models agents as objects, collections of agents as contexts and the relationships between agents as projections. The framework core consists of (1) a data transferring layer, which provides a unified interface for the data transferring calls, (2) distributed containers, which hold instances of agents and sites, (3) projectors that specify relationships between objects in distributed containers, (4) a load balancing layer, which aims to minimize communications between processes while keeping workload distribution even, and (5) I/O interfaces, which implement basic interfaces for loading and storing information about synthetic population and environment. Amos' focus is on reducing modeller efforts related to tuning data exchange and agents' synchronization. As a result, it requires lower levels of C++ and parallel programming proficiency compared to RepastHPC. Moreover, in contrast to Repast HPC, the Amos framework follows the graph-based approach to workload distribution described in D3.3 [14], whereas Repast HPC implements less accurate load balancing techniques.

In Amos, the command line can be used to specify the configuration of a simulation. In this case, the configuration parameters should be listed in a command line using a common Unix-like format for arguments: `--parameter=value`. The names and meanings of these parameters are the same as the parameters from the Pandora configuration files (`config.xml`). In particular, `--num-steps` stands for the number of simulation steps, `--serialize-resolution` stands for the serialization step, etc.

Amos consumes input files in CSV and HDF5 formats. The HDF5 support is implemented by PSNC as a plugin to the core Amos functionality. In case of HDF5 format, synthetic population and synthetic network data must be collected in the same file, following the format described in D4.2 [4]. The input HDF5 file can also encode information about rasters and initial distribution of the synthetic population between processes. In case of CSV format, synthetic population and synthetic network are decoupled in two separate files. Optionally, the user can specify a CSV file with the initial distribution of the synthetic population between processes.

5.2 Data analytics

The data analysis section is partitioned into two use cases. Small-scale (in-memory) analysis, and large-scale (distributed) analysis.

5.2.1 In-memory analysis

R¹⁶ is both a programming language and an environment for statistical computing. It is interpreted (contra compiled) and licensed under the GNU General Public License v2¹⁷ and has support for all major operating systems. While R usage is widespread for doing statistical analysis, a commonly acknowledged shortcoming of R is its execution speed and memory efficiency. Thus, for data sets larger than what can comfortably fit in working memory, R enjoys less usage. Data is typically loaded by reading files into working memory. Thus, file storage access is needed.

RStudio Server¹⁸ is a web IDE (Integrated Development Environment) that support running and editing R code, showing plots, etc. RStudio Server is a free open-source product from the company named RStudio, licensed under Affero General Public License v3¹⁹ (AGPLv3). There exists a commercial alternative license.

¹⁶ <https://www.r-project.org/>

¹⁷ <https://www.gnu.org/licenses/old-licenses/gpl-2.0.en.html>

¹⁸ <https://www.rstudio.com/products/rstudio/>

¹⁹ <https://www.gnu.org/licenses/agpl-3.0.en.html>

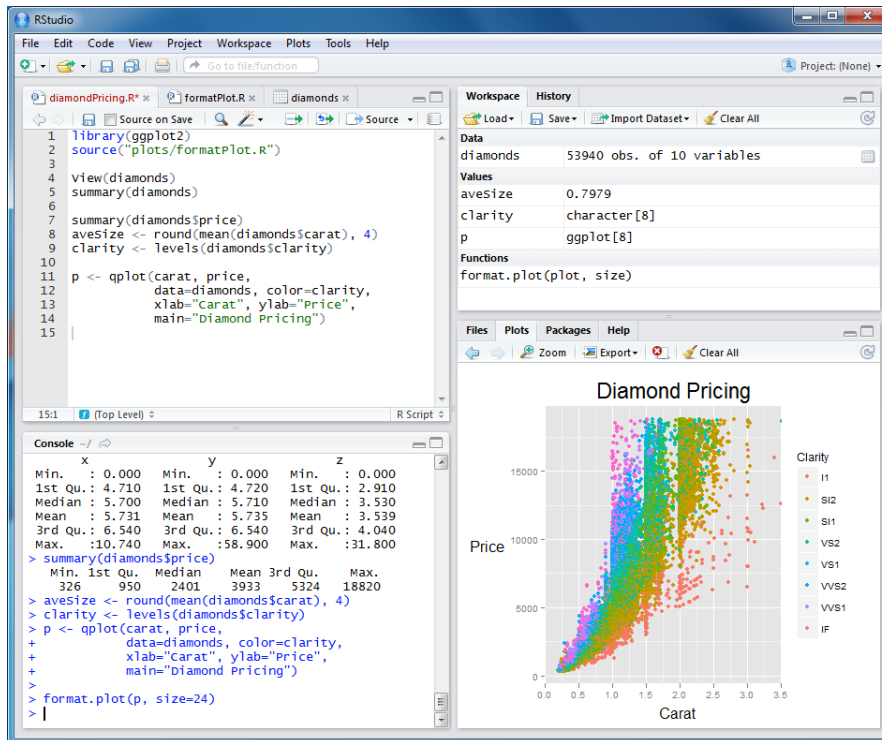


Figure 18. RStudio Server example²⁰.

For the free open source edition, an additional service layer may be necessary to handle sessions and monitor processes, handling login/authentication, etc. These features are provided in the commercial version (“Professional Edition”) for a yearly license fee.

Shiny²¹ is another product from RStudio, also licensed under AGPLv3, with which one can produce interactive data visualizations for the web using R. You can either create a more traditional web page with Shiny components or you can create an interactive document (HTML and JavaScript). Either way, the visualizations are backed by a server which manages the page. The server is expressed as an R function. The user-facing web server must therefore support running an R process in addition to serving the web page.

²⁰ <https://www.rstudio.com/wp-content/uploads/2014/04/rstudio-windows.png>

²¹ <https://shiny.rstudio.com/>

New Orleans Historic and Forecasted Employment Numbers, by Industry

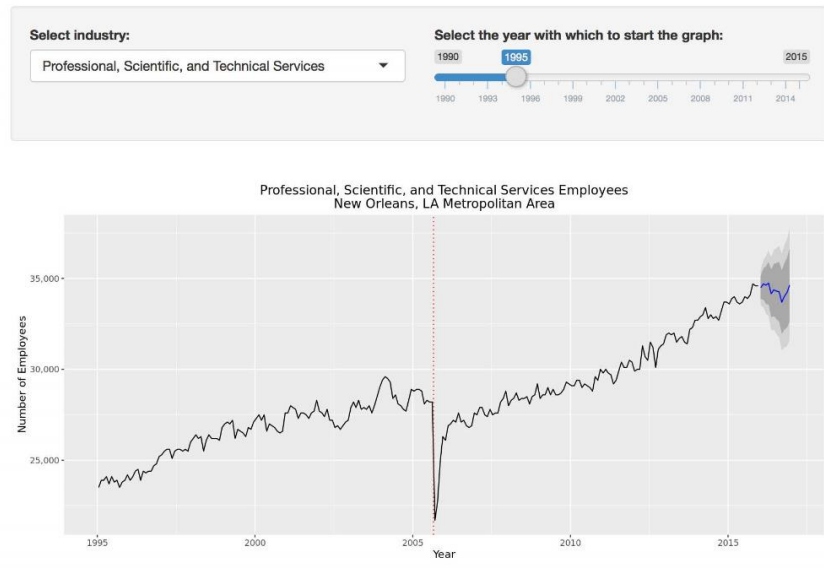


Figure 19. Shiny example²².

The AGPLv3 license, used for RStudio Server and Shiny, requires that the source code of any derivative work based on RStudio Server or Shiny must be made available to any user of the work.

5.2.2 Demanding analysis

If we think of interfaces for demanding analysis, one of the most powerful and comprehensive tools available today is Apache Zeppelin.

Apache Zeppelin²³ is an open-source, web-based “notebook” that enables interactive data analytics and collaborative documents. The notebook is integrated with distributed, general-purpose data processing systems such as Apache Spark (large-scale data processing), Apache Flink (stream processing framework), and many others.

²² <https://static1.squarespace.com/static/54ad9bf4e4b0618d6af9be81/t/56afca0b3b0be33a1d3b443e/1454361388681/>

²³ <https://zeppelin.apache.org/>

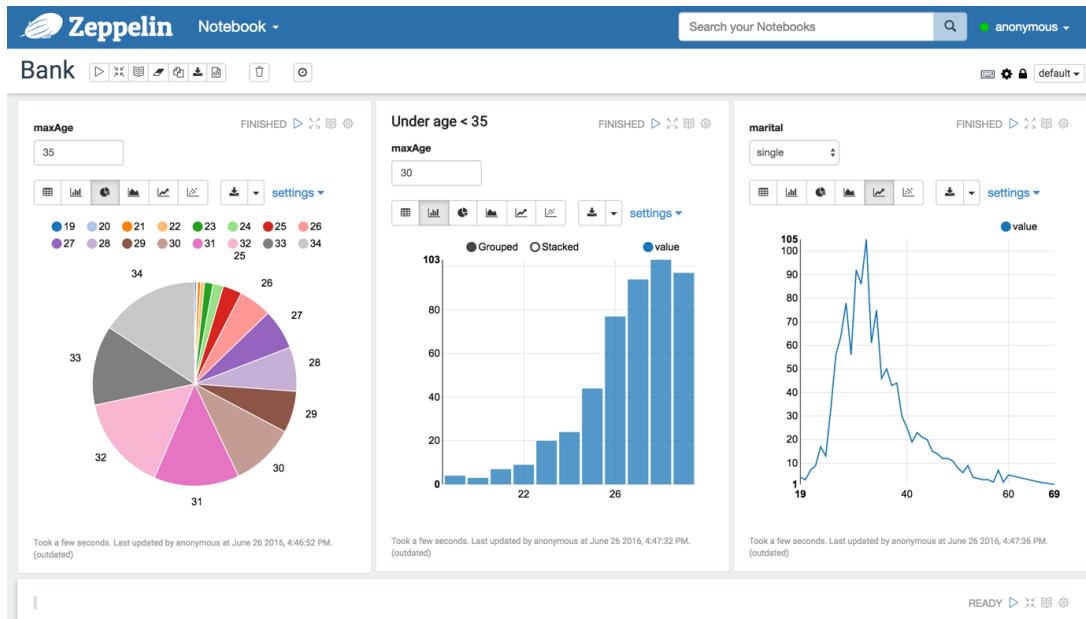


Figure 20. Apache Zeppelin multi-purpose notebook²⁴

It supports multiple languages with an interpreter framework. Apache Zeppelin interpreter²⁵ concept allows any language/data-processing-backend to be plugged into Zeppelin. Currently Apache Zeppelin supports many interpreters such as Apache Spark, Python, JDBC, Markdown and Shell. Currently, it supports interpreters such as Spark, Markdown, Shell, Hive, Phoenix, Tajo, Flink, Ignite, Lens, HBase, Cassandra, Elasticsearch, Geode, PostgreSQL, and Hawq. It can be used for data ingestion, discovery, analytics, and visualizations using notebooks much like to IPython Notebooks. Zeppelin notebooks apprehend output from any language and visualize these using the same tools.

By means an interactive interface Zeppelin offers to you a global view of your analytical results. Apache Zeppelin allows you to make beautiful, data-driven, interactive documents with SQL, Scala, R, or Python right in your browser. Zeppelin offers the possibility of creating a notebook from a web browser and experimenting with the different graphic components it has to create wonderful visualizations that give an added value to analytical results.

Your notebook can be shared among collaborators. Then Apache Zeppelin will broadcast any changes in real-time, just like the collaboration in Google docs.

In the context of CoeGSS project could apply covering the use cases that require a demanding big data due to the common functional pattern in them would be something like: there is a dataset, it is necessary to do something with the data and finally the user obtains comprehensible data using some high level visualization paradigm.

²⁴ <http://zeppelin.apache.org/assets/themes/zeppelin/img/notebook.png>

²⁵ <http://zeppelin.apache.org/docs/latest/development/writingzeppelininterpreter.html#make-your-own-interpreter>

6 Summary

In the deliverable we presented the current status of tools and methods in development that are intended to be implemented in release 3 of the portal.

The primary goal of this document is to present the interoperability aspect of the portal and HPC/HPDA environments. That is why we focused mostly on interfacing dilemmas while the description of the internal functionality development was presented concisely as it is not the core interest of this report.

Many tools developed in the project like AMOS, Synthetic Population or Network Reconstruction are highly innovative solutions and need time and consideration to be implemented from scratch in a way that meets expectations in terms of functionality and scalability. Another important demand is coupling them with the portal interface.

In many inventive systems, first ideas must be verified in practice and modified/adjusted multiple times before an acceptable solution will be worked out. Therefore it must be assumed that information described in this deliverable presents the current state of the knowledge in the project which evolves with the development of the system.

During the next months, verification of proposed ideas and further advances of the practical aspect of integration will be introduced and finally described in deliverable D3.7 in month M36 of the project lifetime.

References

- [1] P. Jansson, M. Lawenda, E. Richter, U. Woessner, C. Ionescu, M. Pařka, R. Schneider, D. Dubhashi, M. Tizzoni, D. Paolotti, S. Fürst, and M. Edwards. *D3.2 – Specification of new method, tools and mechanisms proposed*. Technical report, CoeGSS, 2016.
- [2] E. Richter, W. Schotte, C. Ionescu, R. Schneider, D. Dubhasi, and M. Lawenda. *D3.1 – Available methods, tools and mechanisms*. Technical report, CoeGSS, 2016.
- [3] S. Wolf, D. Paolotti, T. Michele, M. Edwards, S. Fürst, A. Geiges, A. Ireland, F. Schütze, and G. Steudle. *D4.1 – 1st report on pilot requirements*. Technical report, CoeGSS, 2016.
- [4] M. Edwards, S. Fürst, A. Geiges, L. Rossi, M. Tizzoni, and E. Ubaldi. *D4.2 – 2nd report on pilot requirements*. Technical report, CoeGSS, TBD.
- [5] S. Fürst. *Comparing Repast HPC/Pandora*. Technical report, CoeGSS, 2016.
- [6] R. Schneider, S. Gogolenko, M. Gienger, and B. Koller. *CoeGSS – ABM Software stack*. Technical report, CoeGSS, 2016.
- [7] X. Rubio-Campillo. *Pandora: A Versatile Agent-Based Modelling Platform for Social Simulation*. International Conference on Advances in System Simulation – SIMUL, 2014, pp.29-34. ISBN: 978-1-61208-371-1
- [8] J.T. Murphy. *High Performance Agent-Based Modeling in Repast HPC*. Seminar, Argonne, 2014. URL <https://www.ci.uchicago.edu/events/high-performance-agent-based-modeling-repast-hpc>.
- [9] *Grant Agreement No. 676547: CoeGSS*. Report, European Commission, 2015
- [10] M. Gienger, N. Meyer, S. Petruczynik, R. Januszewski, A. Cheptsov, B. Koller. *D5.1 – Definition of the CoeGSS Operation Environment*. Technical report, CoeGSS, 2015
- [11] M. Gienger, B. Karaboga, P. Wolniewicz. *D5.2 – First Operation Report*. Technical report, CoeGSS, 2016.
- [12] F. J. Nieto, B. Karaboga, M. Gienger, A. Rivetti. *D5.9 – Initial Portal Design*. Technical Report, CoeGSS, 2016.
- [13] F. J. Nieto, B. Karaboga, M. Gienger, P. Wolniewicz. *D5.10 – First Portal Release*. Technical report, CoeGSS, 2016.
- [14] P. Jansson, et al. *D3.3 – Second specification of new methods, tools and mechanisms proposed for the support of the application user and programmer*. Technical report, CoeGSS, 2017.

- [15] R. Lambiotte, V.D. Blondel, C. de Kerchove, E. Huens, C. Prieur, Z. Smoreda, and P. Van Dooren, Geographical dispersal of mobile communication networks, *Physica A*, 2008
- [16] M. Barthelemy, *Spatial Networks*, *Physical Review*, 2011
- [17] D. Lin, An Information-Theoretic Definition of Similarity, *ICML '98 Proceedings of the Fifteenth International Conference on Machine Learning*, 1998
- [18] I.T. Jolliffe, *Principal Component Analysis*, Series: Springer Series in Statistics, 2002
- [19] <http://www.stat.cmu.edu/~spew/>
- [20] F. J. Nieto, B. Karaboga, M. Gienger, P. Wolniewicz. D5.11 – Second Portal Release. Technical report, CoeGSS, 2017.

List of tables

Table 1. Visualization requirements.....	26
Table 2. Legend for table of visualization requirements.	27
Table 3. Conversion time to hdf5 output format.....	30

List of figures

Figure 1. Writing into (l.) and reading from (r.) COVISE binary data (RWCovise).....	18
Figure 2 Distributed COVISE Session	20
Figure 3. OpenCOVER SnapShot and Video Plugin	21
Figure 4. Covise net-file (example 1).....	22
Figure 5. Opencover output – visualisation of a time sequence (example 1)	22
Figure 6. Visualisation of specific parameters over time (example 2).....	22
Figure 7. Height displacement of a specific parameter (example 3)	23
Figure 8. Comparing height maps (example 4)	23
Figure 10. Height displacement in perspective projection (example 5)	24
Figure 9. Height displacement in orthographic projection (example 5).....	24
Figure 11. Volume rendering of a time sequence (example 6).....	24
Figure 12. The original CSV and HDF5 file in CKAN dataset.....	28
Figure 13. The data converted from the CSV file.....	28
Figure 14. The data converted from the CSV file.....	29
Figure 15. Conversion time to HDF5 format and processing time of the CKAN database vs file size.....	30
Figure 16. Conversion time to HDF5 format and processing time of the CKAN database vs number of lines	31
Figure 17. The default output from h5serv.....	32
Figure 18. RStudio Server example.....	39
Figure 19. Shiny example.....	40
Figure 20. Apache Zeppelin multi-purpose notebook	41